Claims

What is claimed is:

1. A method of priority queue dispatching in a data processing system, wherein:
a critical section of code for a task is when the task has at least one of a plurality of locks locked; and
said method comprises:
A) dispatching a first task for execution at a temporary dispatching priority higher than a standard dispatching priority for that first task whenever the first task is in a critical section of code for that first task, wherein the critical section of code for that first task is a result of that first task having locked a first lock without having unlocked the first lock; and
B) dispatching the first task for execution at the standard dispatching priority for that first task whenever the first task is not in the critical section of code for that first task.

2. The method in claim 1 wherein:
the method further comprises:
C) queuing a second task in a first FIFO queue when the second task attempts to lock the first lock and fails, wherein:
the FIFO queue comprises a set of tasks waiting to lock the first lock; and
step (C) comprises:
1) placing the second task at an end of the first FIFO queue;
2) comparing the temporary dispatching priority for a third task that is ahead of the second task in the FIFO queue to the temporary dispatching priority of the second task; and
3) setting the temporary dispatching priority of the third task to the temporary dispatching priority of the second task when the temporary dispatching priority of the second task is determined in substep (2) to be greater than the temporary dispatching priority of the third task.

3. The method in claim 2 wherein:
step (C) further comprises:
4) setting the temporary dispatching priority for the second task to the standard dispatching priority for the second task before substep (2) and (3).

4. The method in claim 1 wherein:
the method further comprises:
C) queuing a second task in a first FIFO queue when the second task attempts to lock the first lock and fails, wherein:
the FIFO queue comprises a set of tasks waiting to lock the first lock; and
step (C) comprises:
1) placing the second task at an end of the first FIFO queue;
2) comparing the temporary dispatching priority for the first task to the temporary dispatching priority of the second task; and
3) setting the temporary dispatching priority of the first task to the temporary dispatching priority of the second task when the temporary dispatching priority of the second task is determined in substep (2) to be greater than the temporary dispatching priority of the first task.

5. The method in claim 4 wherein step (C) further comprises:

4) testing whether the first task is in a second FIFO queue awaiting a chance to lock a second lock; and

5) upgrading the temporary dispatching priority of a third task that has a second lock locked when the first task is determined in substep (4) to be in the second FIFO queue awaiting a chance to lock the second lock, wherein substep (5) comprises:

a) comparing the temporary dispatching priority for the third task to the temporary dispatching priority of the second task; and

b) setting the temporary dispatching priority of the third task to the temporary dispatching priority of the second task when the temporary dispatching priority of the second task is determined to be greater than the temporary dispatching priority of the third task.

6. The method in claim 5 wherein:

the method further comprises:

D) comparing a highest temporary dispatching priority of any task in the first FIFO queue to the temporary dispatching priority of the first task after the first task unlocks the second lock; and

E) setting the temporary dispatching priority of the first task to the highest temporary dispatching priority of any task in the first FIFO queue when the temporary dispatching priority of the first task is determined in step (D) to be greater than the highest temporary dispatching priority of any task in the first FIFO queue.

7. The method in claim 4 wherein:
step (c) further comprises:
4) comparing the temporary dispatching priority for a third task that is ahead of the second task in the first FIFO queue to the temporary dispatching priority of the second task; and
5) setting the temporary dispatching priority of the third task to the temporary dispatching priority of the second task when the temporary dispatching priority of the second task is determined in substep (4) to be greater than the temporary dispatching priority of the third task.

8. The method in claim 1 wherein:
the temporary dispatching priority for the first task is determined dynamically and adjusted dynamically depending on the temporary dispatching priority for other tasks behind the first task in a first FIFO queue associated with the first lock.

9. The method in claim 1 wherein:
the temporary dispatching priority for the first task is a system wide value.

10. The method in claim 1 wherein:
the temporary dispatching priority for the first task is a value associated with the first lock when the first lock is a last lock that the first task has attempted to lock.

11. Software stored in a Computer Software Storage Medium for priority queue dispatching in a data processing system, wherein:
a critical section of code for a task is when the task has at least one of a plurality of locks locked; and
said software comprises:
A) dispatching a first task for execution at a temporary dispatching priority higher than a standard dispatching priority for that first task whenever the first task is in a critical section of code for that first task, wherein the critical section of code for that first task is a result of that first task having locked a first lock without having unlocked the first lock; and
B) dispatching the first task for execution at the standard dispatching priority for that first task whenever the first task is not in the critical section of code for that first task.

12. The software in claim 11 wherein:
the software further comprises:
C) queuing a second task in a first FIFO queue when the second task attempts to lock the first lock and fails, wherein:
the FIFO queue comprises a set of tasks waiting to lock the first lock; and
set (C) comprises:
1) placing the second task at an end of the first FIFO queue;
2) comparing the temporary dispatching priority for a third task that is ahead of the second task in the FIFO queue to the temporary dispatching priority of the second task; and
3) setting the temporary dispatching priority of the third task to the temporary dispatching priority of the second task when the temporary dispatching priority of the second task is determined in subset (2) to be greater than the temporary dispatching priority of the third task.

13. The software in claim 12 wherein:
set (C) further comprises:
4) setting the temporary dispatching priority for the second task to the standard dispatching priority for the second task before subset (2) and (3).

14. The software in claim 11 wherein:
the software further comprises:
C) queuing a second task in a first FIFO queue when the second task attempts to lock the first lock and fails, wherein:
the FIFO queue comprises a set of tasks waiting to lock the first lock; and
set (C) comprises:
1) placing the second task at an end of the first FIFO queue;
2) comparing the temporary dispatching priority for the first task to the temporary dispatching priority of the second task; and
3) setting the temporary dispatching priority of the first task to the temporary dispatching priority of the second task when the temporary dispatching priority of the second task is determined in subset (2) to be greater than the temporary dispatching priority of the first task.

15. The software in claim 14 wherein set (C) further comprises:

4) testing whether the first task is in a second FIFO queue awaiting a chance to lock a second lock; and

5) upgrading the temporary dispatching priority of a third task that has a second lock locked when the first task is determined in subset (4) to be in the second FIFO queue awaiting a chance to lock the second lock, wherein subset (5) comprises:

a) comparing the temporary dispatching priority for the third task to the temporary dispatching priority of the second task; and

b) setting the temporary dispatching priority of the third task to the temporary dispatching priority of the second task when the temporary dispatching priority of the second task is determined to be greater than the temporary dispatching priority of the third task.

16. The software in claim 15 wherein:

the software further comprises:

D) comparing a highest temporary dispatching priority of any task in the first FIFO queue to the temporary dispatching priority of the first task after the first task unlocks the second lock; and

E) setting the temporary dispatching priority of the first task to the highest temporary dispatching priority of any task in the first FIFO queue when the temporary dispatching priority of the first task is determined in set (D) to be greater than the highest temporary dispatching priority of any task in the first FIFO queue.

17. The software in claim 14 wherein:
set (c) further comprises:
4) comparing the temporary dispatching priority for a third task that is ahead of the second task in the first FIFO queue to the temporary dispatching priority of the second task; and
5) setting the temporary dispatching priority of the third task to the temporary dispatching priority of the second task when the temporary dispatching priority of the second task is determined in subset (4) to be greater than the temporary dispatching priority of the third task.

18. The software in claim 11 wherein:
the temporary dispatching priority for the first task is determined dynamically and adjusted dynamically depending on the temporary dispatching priority for other tasks behind the first task in a first FIFO queue associated with the first lock.

19. The software in claim 11 wherein:
the temporary dispatching priority for the first task is a system wide value.

20. The software in claim 11 wherein:
the temporary dispatching priority for the first task is a value associated with the first lock when the first lock is a last lock that the first task has attempted to lock.

21. A computer readable Non-Volatile Storage Medium encoded with software for priority queue dispatching in a data processing system, wherein:

a critical section of code for a task is when the task has at least one of a plurality of locks locked; and

said software comprises:

A) dispatching a first task for execution at a temporary dispatching priority higher than a standard dispatching priority for that first task whenever the first task is in a critical section of code for that first task, wherein the critical section of code for that first task is a result of that first task having locked a first lock without having unlocked the first lock; and

B) dispatching the first task for execution at the standard dispatching priority for that first task whenever the first task is not in the critical section of code for that first task.